Week 3 — Student Churn Analysis & Predictive Modeling

This notebook builds predictive models for student drop-offs (churn), evaluates performance, analyzes key drivers, and exports a PDF report.

Inputs: `Week 1 Deliverable- Data Cleanup (1).xlsx`
Outputs:

- `Week3_Churn_Analysis_Report.pdf`
- `week3_model_metrics.csv`
- `week3_feature_importance.csv`

Label definition: `churn = 1` if `Status Description` is in `['Withdraw','Dropped Out']`, else `0`.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.metrics import (accuracy_score, precision_score,
recall_score,
                             f1_score, roc_auc_score,
confusion_matrix, roc_curve)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.inspection import permutation_importance

plt.rcParams.update({'figure.dpi': 130})

# Update the path if needed
data_path = "Week 1 Deliverable- Data Cleanup (1).xlsx"
df = pd.read_excel(data_path)
print("Rows, Cols:", df.shape)
display(df.head(3))
```

```
Rows, Cols: (8560, 18)
```

```
{"summary":"{\n  \"name\": \"display(df\",\n  \"rows\": 3,\n
\"fields\": [\n    {\n      \"column\": \"Learner SignUp DateTime\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"06/14/2023 12:30:35\",\n          \"2023-01-05 05:29:16\",\n
\"2023-09-04 20:35:08\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Opportunity Id\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"00000000-0GN2-A0AY-7XK8-C5FZPP\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Opportunity Name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Career Essentials: Getting Started with Your
Professional Journey\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Opportunity Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Course\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Opportunity End Date\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"06/29/2024 18:52:39\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"First Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Faria\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Date of Birth\",\n      \"properties\": {\n        \"dtype\":
\"date\",\n        \"min\": \"2000-08-16 00:00:00\",\n        \"max\":
\"2002-01-27 00:00:00\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"2001-12-01 00:00:00\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Gender\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Pakistan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Institution Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Nwihs\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Current/Intended Major\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 3,\n
```

\"samples\": [\n   \"Radiology\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Entry created at\",\n
\"properties\": {\n   \"dtype\": \"date\",\n   \"min\":
\"2024-11-03 12:01:41\",\n   \"max\": \"2024-11-03 12:01:41\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n   \"2024-
11-03 12:01:41\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Status Description\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   \"Started\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Status Code\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.0,\n   \"min\":
1080.0,\n   \"max\": 1080.0,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   1080.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Apply Date\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n
\"06/14/2023 12:36:09\"\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Opportunity Start Date\",\n   \"properties\": {\n
\"dtype\": \"date\",\n   \"min\": \"2022-03-11 18:30:39\",\n
\"max\": \"2022-03-11 18:30:39\",\n   \"num_unique_values\": 1,\n
\"samples\": [\n   \"2022-03-11 18:30:39\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1.1547005383792515,\n
\"min\": 23.0,\n   \"max\": 25.0,\n   \"num_unique_values\":
2,\n   \"samples\": [\n   25.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Opportunity Duration (Days)\",\n
\"properties\": {\n   \"dtype\": \"date\",\n   \"min\":
841,\n   \"max\": 841,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   841\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe"}

```
# Keep known statuses and define churn label
valid_statuses = ['Withdraw', 'Dropped Out', 'Applied', 'Started',
'Team Allocated', 'Waitlisted', 'Rewards Award', 'Rejected']
df = df[df['Status
Description'].astype(str).isin(valid_statuses)].copy()
df['churn'] = df['Status
Description'].astype(str).isin(['Withdraw','Dropped Out']).astype(int)

# Parse dates if present
for c in ['Learner SignUp DateTime','Opportunity End Date','Date of
```

```
Birth','Apply Date','Opportunity Start Date','Entry created at']:
    if c in df.columns:
        df[c] = pd.to_datetime(df[c], errors='coerce')

# Engineered features
df['Age_at_Apply'] = np.where(df['Date of Birth'].notna() & df['Apply
Date'].notna(),
                              (df['Apply Date'] - df['Date of
Birth']).dt.days/365.25, np.nan)
df['Apply_to_Start_Days'] = (df['Opportunity Start Date'] - df['Apply
Date']).dt.days
df['Start_to_End_Days'] = (df['Opportunity End Date'] -
df['Opportunity Start Date']).dt.days
df['Signup_to_Apply_Days'] = (df['Apply Date'] - df['Learner SignUp
DateTime']).dt.days

# Columns to exclude
drop_cols = ['Opportunity Id','First Name','Entry created at','Status
Code',
             'Learner SignUp DateTime','Opportunity End Date','Date of
Birth','Apply Date','Opportunity Start Date']

features = [c for c in df.columns if c not in drop_cols +
['churn','Status Description']]
X = df[features].copy()
y = df['churn'].astype(int)

# Split numeric/categorical
num_cols =
X.select_dtypes(include=['int64','float64','int32','float32']).columns
.tolist()
cat_cols = [c for c in X.columns if c not in num_cols]

# Coerce categoricals to strings
for c in cat_cols:
    X[c] = X[c].astype(str)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42, stratify=y)

num_tf = Pipeline([('imputer', SimpleImputer(strategy='median')),
('scaler', StandardScaler())])
cat_tf = Pipeline([('imputer',
SimpleImputer(strategy='most_frequent')), ('onehot',
OneHotEncoder(handle_unknown='ignore'))])
preprocess = ColumnTransformer([('num', num_tf, num_cols), ('cat',
cat_tf, cat_cols)])


models = {
```

```
    "Logistic Regression": LogisticRegression(max_iter=500,
class_weight="balanced"),
    "Random Forest": RandomForestClassifier(n_estimators=300,
random_state=42, class_weight="balanced"),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}

results = []
fprs_tprs = {}
trained = {}
best_name, best_auc = None, -1.0

for name, model in models.items():
    pipe = Pipeline([('preprocess', preprocess), ('model', model)])
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    if hasattr(pipe.named_steps['model'], 'predict_proba'):
        y_proba = pipe.predict_proba(X_test)[:,1]
    else:
        dec = pipe.decision_function(X_test)
        y_proba = (dec - dec.min()) / (dec.max() - dec.min() + 1e-9)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, zero_division=0)
    rec = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)
    auc_val = roc_auc_score(y_test, y_proba)
    results.append({"Model": name, "Accuracy": acc, "Precision": prec,
"Recall": rec, "F1": f1, "ROC_AUC": auc_val})
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    fprs_tprs[name] = (fpr, tpr, auc_val)
    trained[name] = pipe
    if auc_val > best_auc:
        best_auc, best_name = auc_val, name

metrics_df = pd.DataFrame(results).sort_values("ROC_AUC",
ascending=False)
metrics_df
```


```
{"summary":"{\n  \"name\": \"metrics_df\",\n  \"rows\": 3,\n
\"fields\": [\n    {\n      \"column\": \"Model\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Gradient Boosting\",\n          \"Logistic Regression\",\n
\"Random Forest\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Accuracy\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.007244246700457023,\n        \"min\":
0.9616822429906542,\n        \"max\": 0.9752336448598131,\n
\"num_unique_values\": 3,\n        \"samples\": [\n
0.9752336448598131,\n          0.9616822429906542,\n
```

0.9728971962616823\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Precision\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.11027126723552104,\n   \"min\": 0.7008547008547008,\n   \"max\": 0.912751677852349,\n \"num_unique_values\": 3,\n   \"samples\": [\n 0.912751677852349,\n   0.7008547008547008,\n 0.8597560975609756\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Recall\",\n   \"properties\": {\n   \"dtype\": \"number\",\n \"std\": 0.08484763933561405,\n   \"min\": 0.7727272727272727,\n \"max\": 0.9318181818181818,\n   \"num_unique_values\": 3,\n \"samples\": [\n   0.7727272727272727,\n 0.9318181818181818,\n   0.8011363636363636\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"F1\",\n   \"properties\": {\n \"dtype\": \"number\",\n   \"std\": 0.019514034794671124,\n \"min\": 0.8,\n   \"max\": 0.8369230769230769,\n \"num_unique_values\": 3,\n   \"samples\": [\n 0.8369230769230769,\n   0.8,\n   0.8294117647058824\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n }\n   },\n   {\n   \"column\": \"ROC_AUC\",\n \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.001797688730131981,\n   \"min\": 0.9690450842436585,\n \"max\": 0.9725166635808185,\n   \"num_unique_values\": 3,\n \"samples\": [\n   0.9725166635808185,\n 0.9715909090909091,\n   0.9690450842436585\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"metrics_df"}

```
metrics_df.to_csv("week3_model_metrics.csv", index=False)
print("Saved: week3_model_metrics.csv")
best_name, best_auc
```

```
Saved: week3_model_metrics.csv
```

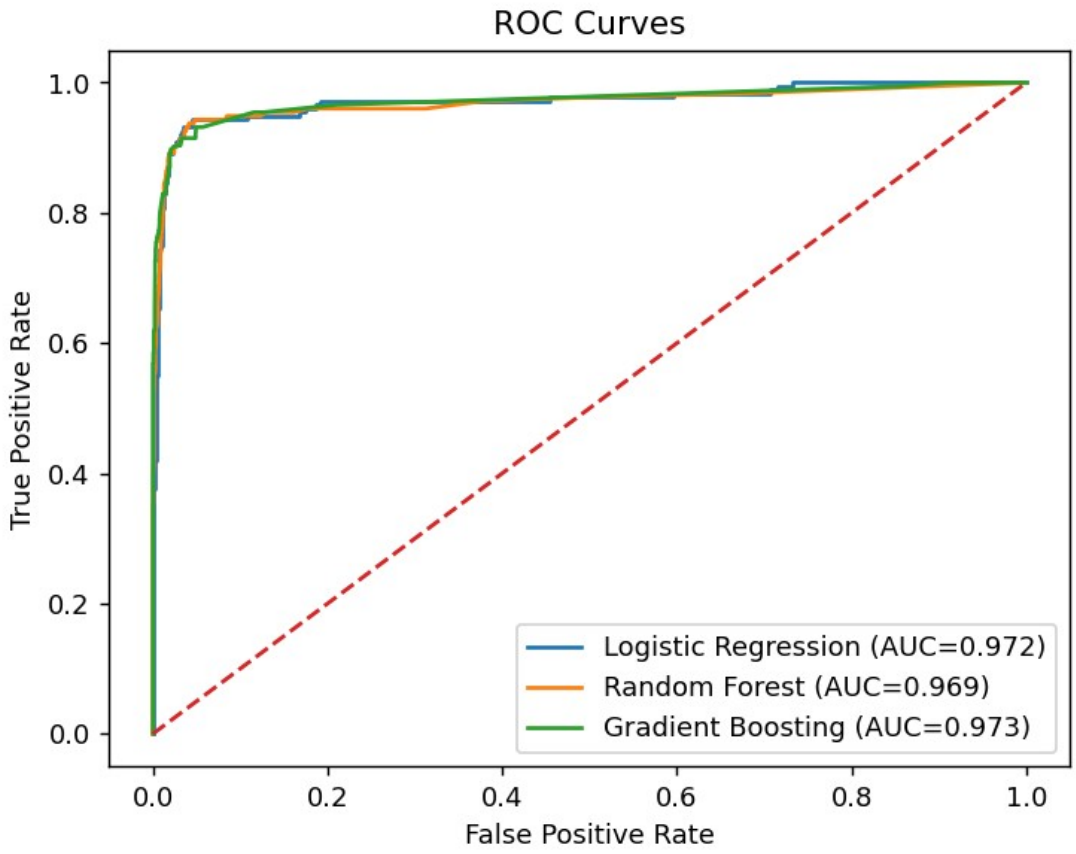
```
('Gradient Boosting', np.float64(0.9725166635808185))
```

```
best_model = trained[best_name]
perm = permutation_importance(best_model, X_test, y_test, n_repeats=5,
random_state=42, scoring='roc_auc')
ohe =
best_model.named_steps['preprocess'].named_transformers_['cat'].named_
steps['onehot']
cat_feature_names = ohe.get_feature_names_out([c for c in X.columns if
c not in
X.select_dtypes(include=['int64','float64','int32','float32']).columns
]).tolist()
```

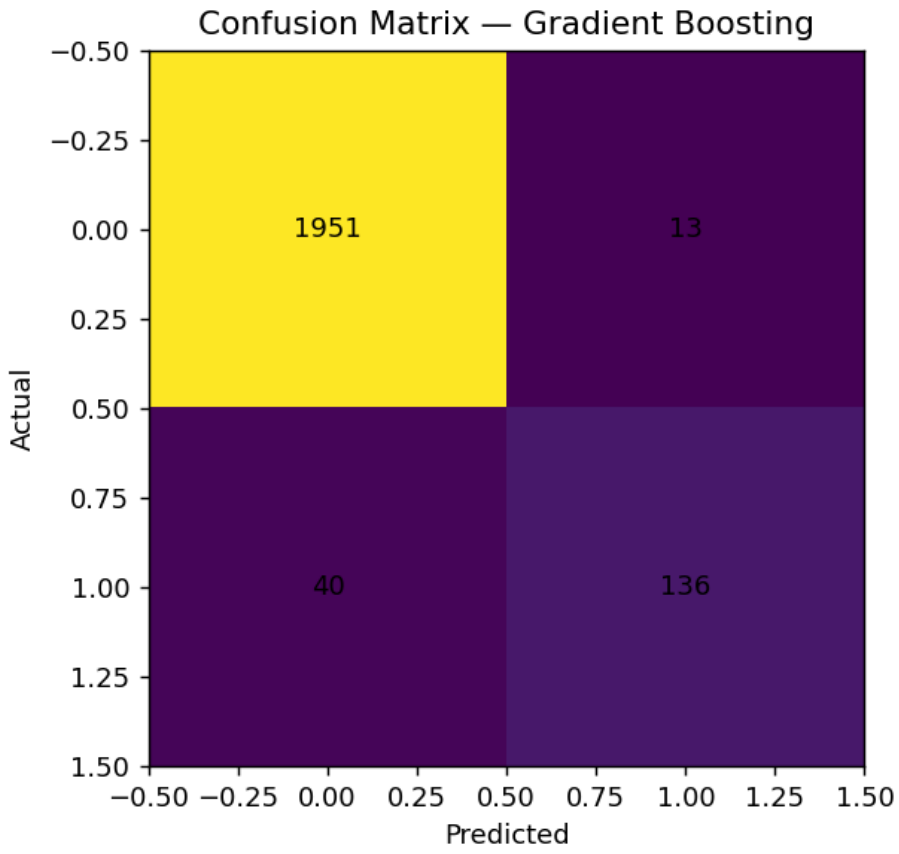
```
num_feature_names =
X.select_dtypes(include=['int64','float64','int32','float32']).columns
.tolist()
feature_names = num_feature_names + cat_feature_names

imp_df = pd.DataFrame({
    "feature": feature_names[:len(perm.importances_mean)],
    "importance_mean": perm.importances_mean[:len(feature_names)],
    "importance_std": perm.importances_std[:len(feature_names)]
}).sort_values("importance_mean", ascending=False)

imp_df.to_csv("week3_feature_importance.csv", index=False)
imp_df.head(20)
```


```
{"summary":"{\n  \"name\": \"imp_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"feature\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 12,\n
\"samples\": [\n          \"Opportunity Name_Data Visualization
Associate\",\n          \"Age_at_Apply\",\n          \"Opportunity
Name_CPR/AED Certification\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"importance_mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.07559280789320241,\n
\"min\": -0.00012092668024434871,\n        \"max\":
0.2638672236622849,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          0.0,\n          0.017494445473060605,\n
0.00010299018700246076\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"importance_std\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.001746022179423105,\n        \"min\":
0.0,\n        \"max\": 0.004780324549275826,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          0.0,\n
0.004223812778914464,\n          0.00010046359131763039\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"imp_df"}
```

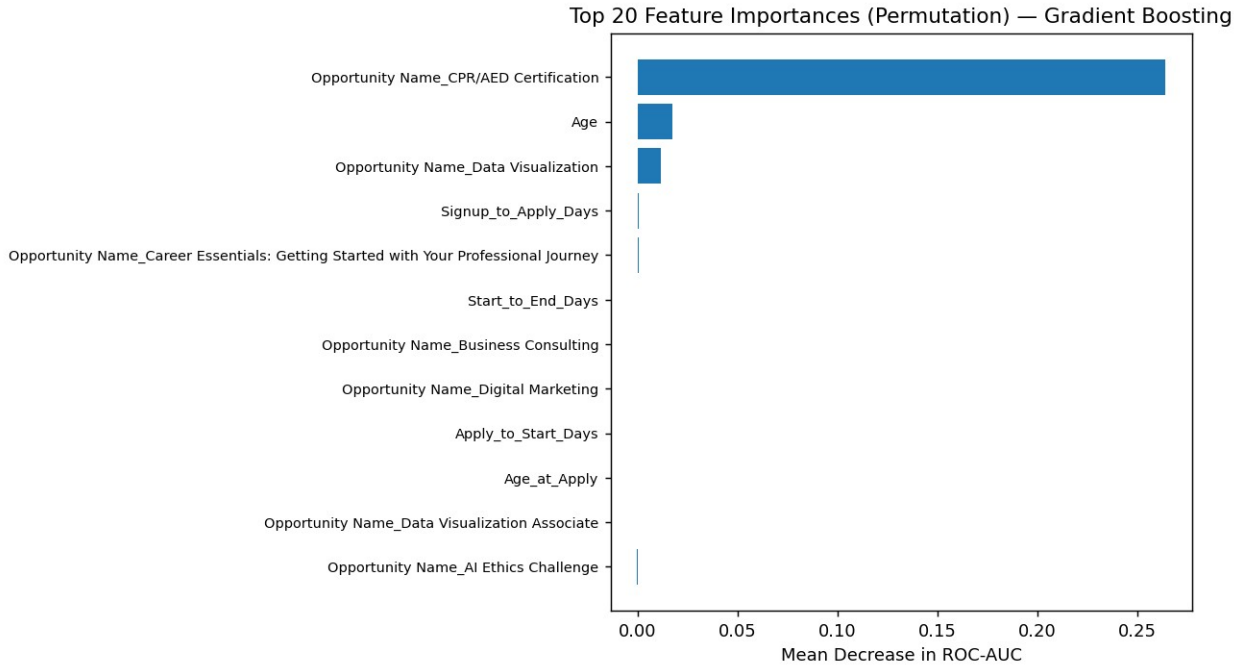

```
# ROC curves
plt.figure()
for name, (fpr, tpr, auc_val) in fprs_tprs.items():
    plt.plot(fpr, tpr, label=f"{name} (AUC={auc_val:.3f})")
plt.plot([0,1],[0,1], linestyle='--')
plt.title("ROC Curves")
plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate");
plt.legend(loc="lower right")
plt.show()
```




```
# Confusion matrix for best model
from sklearn.metrics import confusion_matrix
y_pred_best = best_model.predict(X_test)
cm = confusion_matrix(y_test, y_pred_best)
plt.figure()
plt.imshow(cm, interpolation='nearest')
plt.title(f"Confusion Matrix — {best_name}")
plt.xlabel("Predicted"); plt.ylabel("Actual")
for (i, j), v in np.ndenumerate(cm):
    plt.text(j, i, int(v), ha='center', va='center')
plt.show()
```




```
# Top 20 feature importances
topk = imp_df.head(20)
plt.figure(figsize=(6,6))
plt.barh(range(len(topk)), topk['importance_mean'][::-1])
plt.yticks(range(len(topk)), topk['feature'][::-1], fontsize=8)
plt.title(f"Top 20 Feature Importances (Permutation) — {best_name}")
plt.xlabel("Mean Decrease in ROC-AUC")
plt.show()
```




```
pdf_path = "Week3_Churn_Analysis_Report.pdf"
with PdfPages(pdf_path) as pdf:
    plt.figure(figsize=(8.5, 11)); plt.axis('off')
    pos_rate = y.mean()
    plt.text(0.5, 0.85, "Week 3: Churn Analysis & Predictive
Modeling", ha='center', fontsize=20, weight='bold')
    plt.text(0.5, 0.80, "Student Sign-off | Predictive Models,
Performance & Key Drivers", ha='center', fontsize=12)
    meta = f"Records: {len(df)} | Features used: {len(features)} |
Churn positive rate: {pos_rate:.2%}\n"
    meta += "Label: churn=1 if Status in ['Withdraw','Dropped Out'];
else 0"
    plt.text(0.5, 0.72, meta, ha='center', fontsize=10)
    pdf.savefig(); plt.close()

    plt.figure(figsize=(8.5, 11)); plt.axis('off')
    txt = ("Introduction & Methods\n\n"
           "Objective: Forecast student churn and identify drivers to
inform retention.\n\n"
           "Data Preparation:\n"
           "- Parsed date fields and engineered: Age_at_Apply,
Apply_to_Start_Days, Start_to_End_Days, Signup_to_Apply_Days.\n"
           "- Categorical encoding with One-Hot; numeric imputation
(median) and scaling.\n"
           "- Train/Test split: 75/25 with stratification.\n\n"
           "Models: Logistic Regression (balanced), Random Forest
(balanced), Gradient Boosting.\n"
           "Evaluation Metrics: Accuracy, Precision, Recall, F1, ROC-
```

```
AUC.\n")
    plt.text(0.05, 0.95, txt, ha='left', va='top', fontsize=11,
wrap=True)
    pdf.savefig(); plt.close()

    # Metrics table
    plt.figure(figsize=(8.5, 11)); plt.axis('off')
    plt.text(0.5, 0.95, "Model Performance (Test Set)", ha='center',
va='top', fontsize=16, weight='bold')
    from matplotlib.table import Table
    ax = plt.gca()
    table_data =
[["Model","Accuracy","Precision","Recall","F1","ROC_AUC"]] +
[[row['Model'], f"{row['Accuracy']:.3f}", f"{row['Precision']:.3f}",
f"{row['Recall']:.3f}", f"{row['F1']:.3f}", f"{row['ROC_AUC']:.3f}"]
for _, row in metrics_df.iterrows()]
    tab = Table(ax, bbox=[0.05, 0.1, 0.9, 0.8])
    n_rows, n_cols = len(table_data), len(table_data[0])
    for i in range(n_rows):
        for j in range(n_cols):
            tab.add_cell(i, j, 0.9/n_cols, 0.8/n_rows,
text=table_data[i][j], loc='center')
    ax.add_table(tab); ax.set_xlim(0,1); ax.set_ylim(0,1)
    pdf.savefig(); plt.close()

    # ROC curves
    plt.figure(figsize=(8.5, 11))
    for name, (fpr, tpr, auc_val) in fprs_tprs.items():
        plt.plot(fpr, tpr, label=f"{name} (AUC={auc_val:.3f})")
    plt.plot([0,1],[0,1], linestyle='--')
    plt.title("ROC Curves"); plt.xlabel("False Positive Rate");
plt.ylabel("True Positive Rate"); plt.legend(loc="lower right")
    pdf.savefig(); plt.close()

    # Confusion matrix
    from sklearn.metrics import confusion_matrix
    y_pred_best = best_model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred_best)
    plt.figure(figsize=(6,6))
    plt.imshow(cm, interpolation='nearest')
    plt.title(f"Confusion Matrix — {best_name}")
    plt.xlabel("Predicted"); plt.ylabel("Actual")
    for (i, j), v in np.ndenumerate(cm):
        plt.text(j, i, int(v), ha='center', va='center')
    pdf.savefig(); plt.close()

    # Feature importance
    topk = imp_df.head(20)
    plt.figure(figsize=(8.5, 11))
    plt.barh(range(len(topk)), topk['importance_mean'][::-1])
```

```
    plt.yticks(range(len(topk)), topk['feature'][::-1], fontsize=8)
    plt.title(f"Top 20 Feature Importances (Permutation) —
{best_name}")
    plt.xlabel("Mean Decrease in ROC-AUC")
    pdf.savefig(); plt.close()

print("Saved:", pdf_path)
```

```
Saved: Week3_Churn_Analysis_Report.pdf
```

To upload the dataset:

1. Click the **folder icon** on the left sidebar to open the file browser.
2. Click the **upload icon** (up arrow).
3. Select the file named `Week 1 Deliverable- Data Cleanup (1).xlsx` from your local machine and upload it.

Once the file is uploaded, the previous code cell that failed should now be able to load the data.